# Чекпоинт 2: Core (RAG Pipeline)

RAG Чат-бот по роману «Мастер и Маргарита»

**Команда bookworm**

Горбунов Дмитрий Павлович
Ковалева Дарина Евгеньевна
Тельнов Федор Николаевич
Мацаков Борис Вячеславович

**Репозиторий:** https://github.com/bookworm-itmo/llm-intro-project

## Архитектура пайплайна

Система построена по классической RAG-архитектуре и состоит из четырёх основных компонентов:

- **Data Service** — парсинг FB2, chunking текста, подготовка данных
- **RAG Service** — векторизация запросов, поиск релевантных фрагментов через FAISS
- **LLM Service** — генерация ответов через Claude API
- **Frontend** — веб-интерфейс на Streamlit

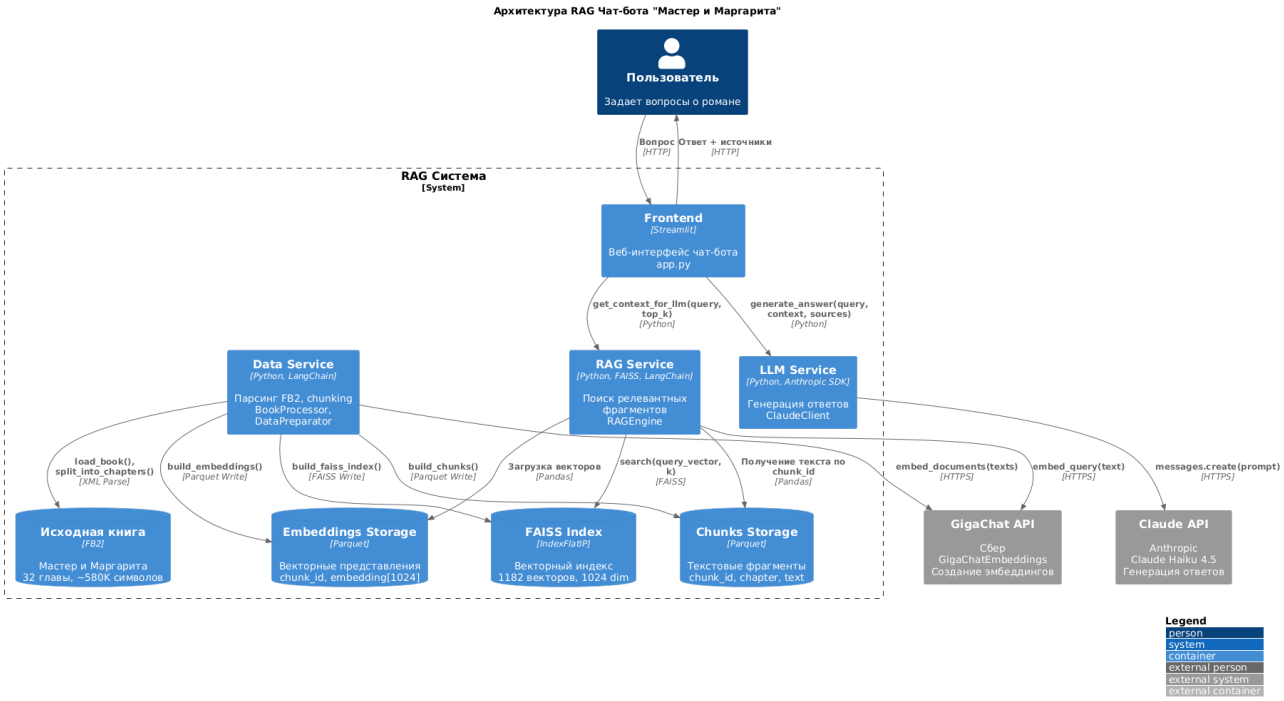



Рис. 1: Архитектура RAG чат-бота

# Взаимодействие компонентов

## Offline-этап: подготовка данных

1. **BookProcessor** парсит FB2-файл и разбивает текст на главы
2. **RecursiveCharacterTextSplitter** создаёт чанки (800 символов, overlap 100)
3. **GigaChatEmbeddings** генерирует векторы размерности 1024
4. **FAISS IndexFlatIP** индексирует нормализованные эмбеддинги

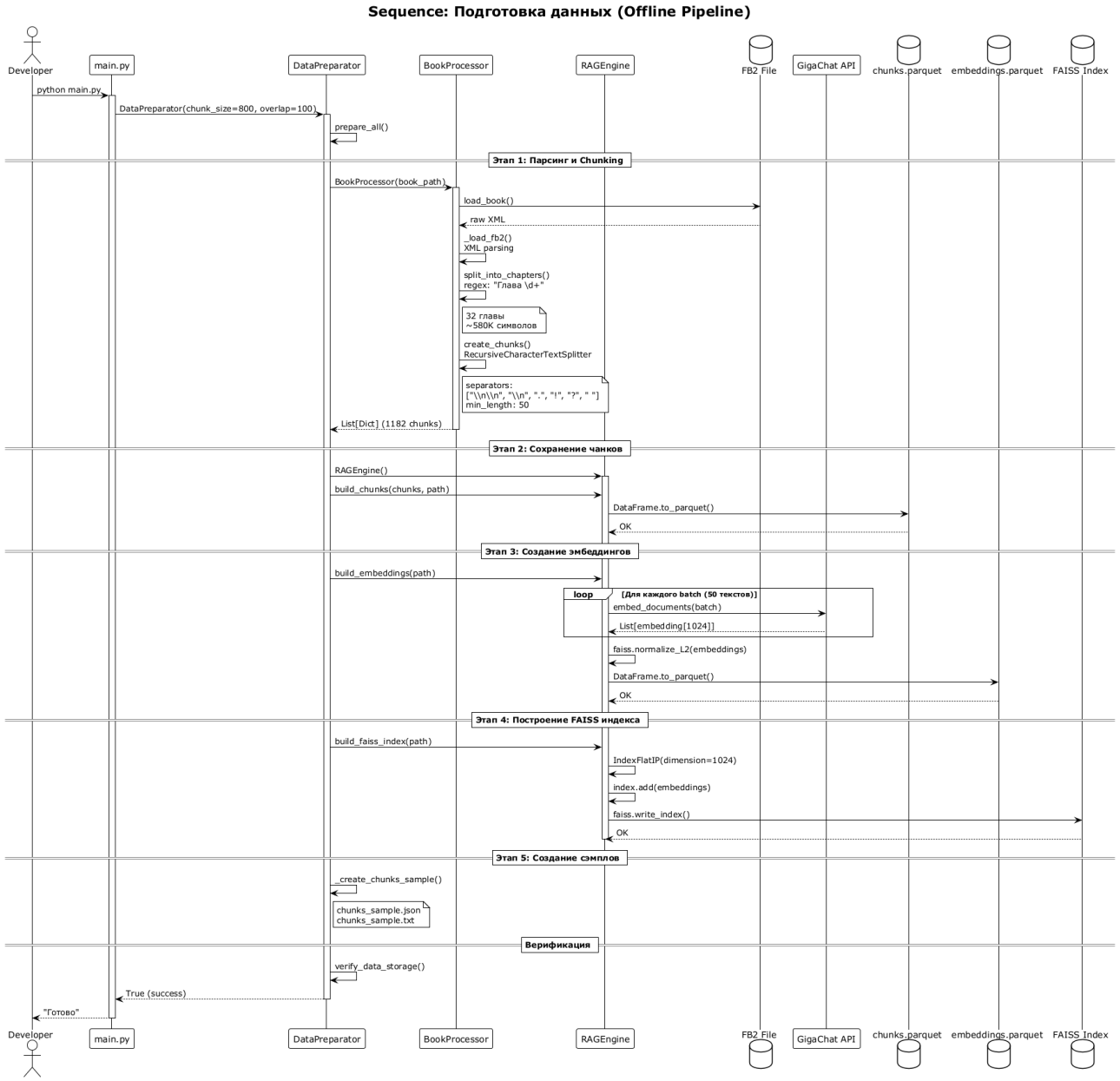



Рис. 2: Sequence-диаграмма: подготовка данных

## Online-этап: обработка запроса

1. Пользователь вводит вопрос через Streamlit UI
2. **RAGEngine.search()** создаёт эмбеддинг запроса через GigaChat API
3. FAISS находит top-k ближайших чанков по косинусному сходству
4. **ClaudeClient.generate_answer()** формирует промпт с контекстом
5. Claude Haiku 4.5 генерирует ответ на основе контекста
6. Ответ и источники отображаются пользователю

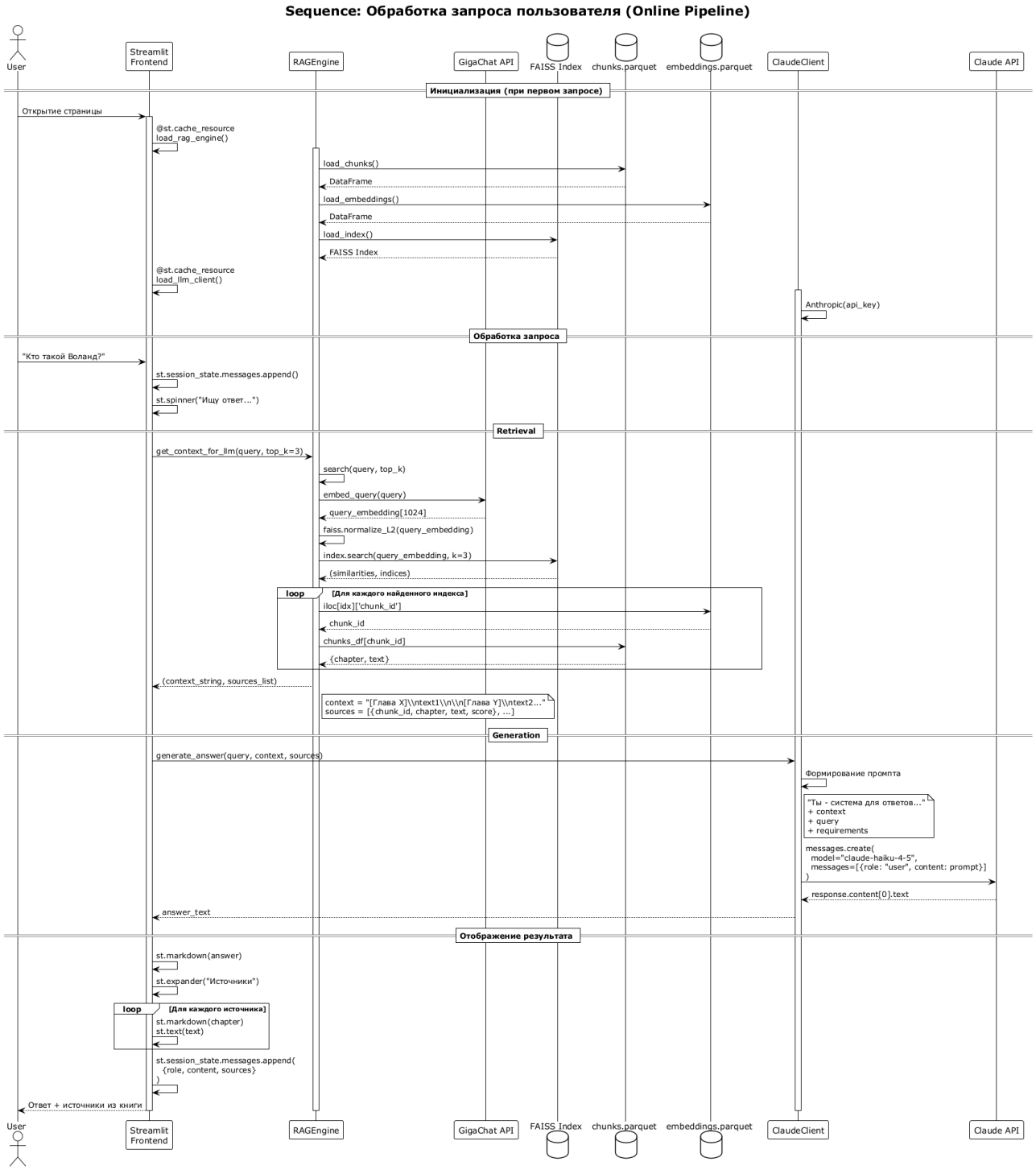



Рис. 3: Sequence-диаграмма: обработка запроса пользователя

# Список зависимостей

| Компонент | Технология |
|---|---|
| Chunking | LangChain RecursiveCharacterTextSplitter |
| Embeddings | GigaChat Embeddings (Сбер), 1024 dim |
| Vector Store | FAISS IndexFlatIP |
| LLM (генерация) | Claude Haiku 4.5 (Anthropic) |
| LLM (eval) | GigaChat-2 |
| Frontend | Streamlit |
| Data Format | Parquet (chunks, embeddings) |

# Реализация основной логики

## RAG Pipeline

**Векторизация:** Используется GigaChatEmbeddings с нормализацией L2 для косинусного сходства.

**База:** FAISS IndexFlatIP — точный поиск ближайших соседей по inner product.

**Retrieval:** Поиск top-k=5 релевантных чанков по запросу.

## End-to-end Pipeline

Запрос $\rightarrow$ Embedding $\rightarrow$ FAISS Search $\rightarrow$ Context Assembly $\rightarrow$ LLM Generation $\rightarrow$ Ответ

Промпт для LLM включает:

- Контекст из найденных чанков с указанием глав
- Инструкцию отвечать только на основе контекста
- Требование указывать источники и цитаты

# Тестирование и метрики

## Валидационная выборка

| | |
|---|---|
| Всего запросов: | 70 |
| Контекст на запрос: | 5 чанков |
| Средний объём контекста: | $\sim$700 символов |

## Пайплайн тестирования

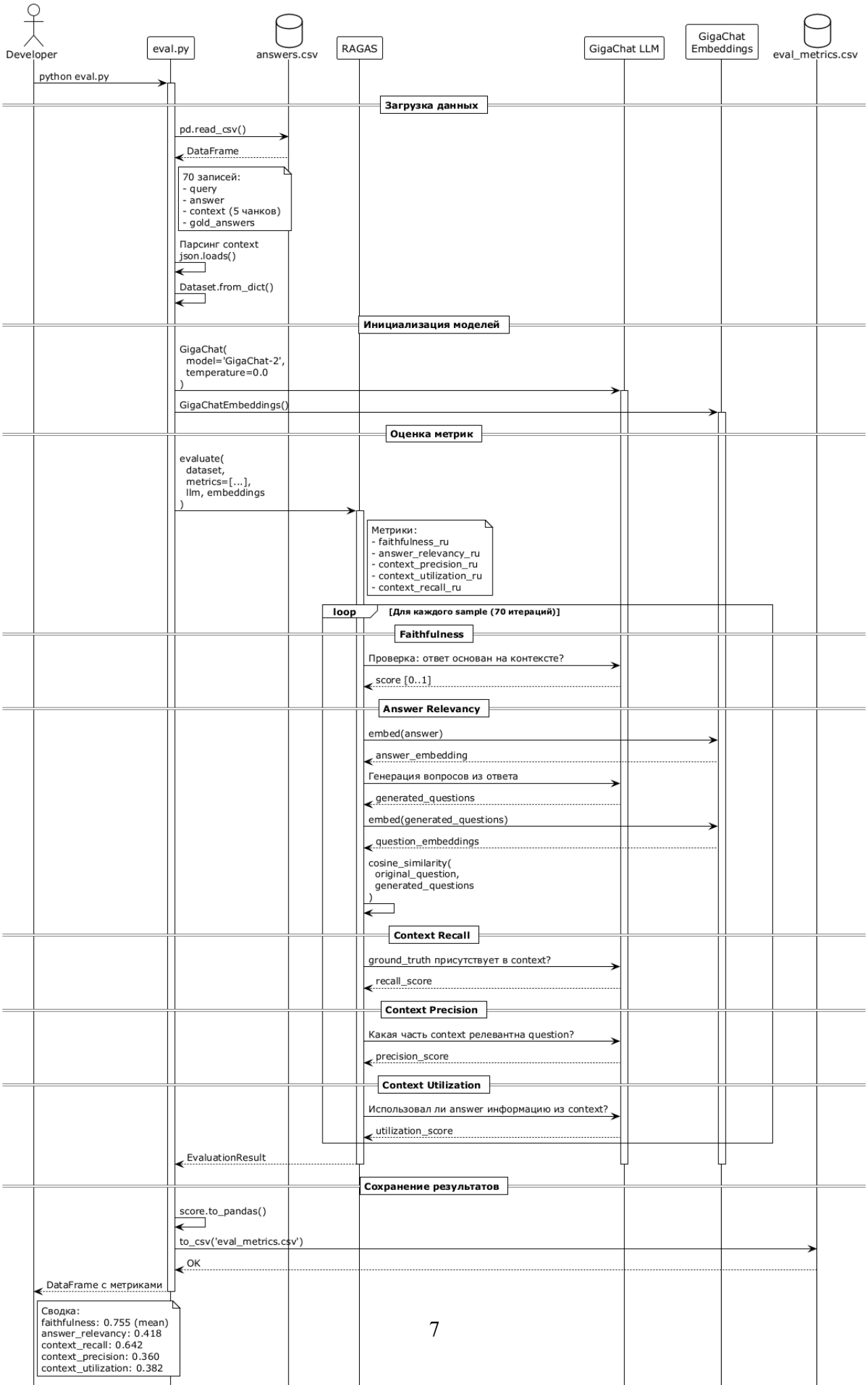
Для оценки качества использовалась библиотека **RAGAS** с адаптацией для русского языка:

```
from ragas import evaluate
```

```
from ragas.metrics import faithfulness_ru, answer_relevancy_ru,
    context_precision_ru, context_utilization_ru, context_recall_ru

score = evaluate(
    dataset=dataset,
    metrics=[faithfulness_ru, answer_relevancy_ru,
             context_precision_ru, context_utilization_ru,
             context_recall_ru],
    llm=GigaChat-2,
    embeddings=GigaChatEmbeddings
)
```

## Sequence: Оценка качества RAG-системы (Evaluation Pipeline)



### Описание метрик

- **faithfulness** — насколько ответ опирается на контекст (меньше галлюцинаций)
- **answer_relevancy** — насколько ответ соответствует вопросу
- **context_recall** — нашёл ли retrieval нужную информацию
- **context_precision** — «чистота» контекста (доля релевантного)
- **context_utilization** — использовала ли модель контекст при ответе

### Результаты (Baseline)

| | faithfulness | answer_rel. | ctx_precision | ctx_util. | ctx_recall |
|---|---|---|---|---|---|
| mean | 0.755 | 0.418 | 0.360 | 0.382 | 0.642 |
| median | 0.764 | 0.000 | 0.287 | 0.287 | 1.000 |
| std | 0.223 | 0.460 | 0.380 | 0.411 | 0.465 |
| min | 0.000 | 0.000 | 0.000 | 0.000 | 0.000 |
| max | 1.000 | 0.976 | 1.000 | 1.000 | 1.000 |

Таблица 1: Сводные метрики качества RAG-системы

### Анализ ошибок

| | |
|---|---|
| Отказы («недостаточно контекста»): | 31 из 70 (44.3%) |
| Полный retrieval (context_recall = 1.0): | 42 запроса |
| Провал retrieval (context_recall < 0.5): | 23 запроса |
| Провал генерации при полном retrieval: | 19 запросов |
| «Ответ есть, но модель отказалась»: | 15 запросов |

## Интерпретация результатов

### Сильные стороны

**Faithfulness (0.755)** — высокий показатель. Когда модель отвечает содержательно, она не «улетает» в галлюцинации и опирается на предоставленный контекст.

### Проблемы Retrieval

Средний context_recall = 0.642 при медиане 1.0 указывает на бимодальное распределение: retrieval либо находит нужную информацию полностью, либо не находит вовсе. 23 запроса с context_recall < 0.5 — потенциальные точки улучшения chunking/embedding стратегии.

### Проблемы генерации

Критичный паттерн: 15 случаев, когда ответ присутствует в контексте (context_recall=1), но модель отказывается отвечать. Это указывает на слишком консервативный промпт — модель предпочитает отказ вместо извлечения информации.

### Направления улучшения (Checkpoint 3)

1. Оптимизация промпта: добавить инструкцию «сначала найди цитату, потом ответь»
2. Увеличение top-k или уменьшение chunk_size для лучшего покрытия
3. Гибридный поиск (BM25 + dense retrieval)
4. Reranking найденных чанков

## Структура репозитория

```
.
├── data/                          # Данные и индексы
│   ├── master_and_margarita.fb2
│   ├── chunks.parquet
│   ├── embeddings.parquet
│   └── faiss_index/index.faiss
├── services/
│   ├── data_service/              # BookProcessor, DataPreparator
│   ├── rag_service/               # RAGEngine (FAISS + GigaChat)
│   └── llm_service/               # ClaudeClient
├── frontend/                      # Streamlit UI
├── metrics/                       # Скрипты и отчёты по метрикам
│   ├── eval.py
│   ├── answers.csv
│   └── report.md
├── validation/                    # Валидационные данные
└── docs/                          # Документация и диаграммы
```